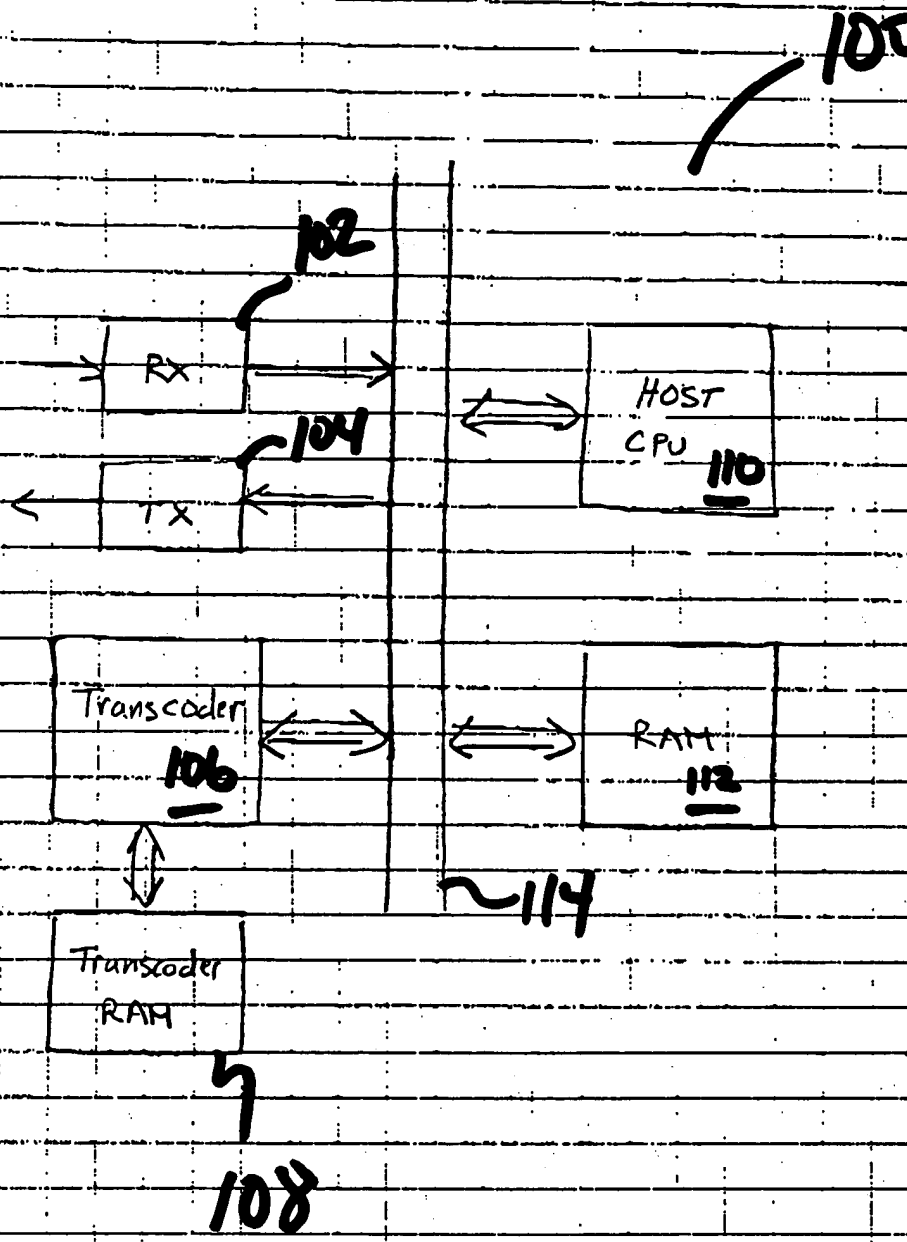

100
102
RX
104
TX
HOST
CPU
110
Transcoder
106
RAM
112
114
Transcoder
RAM
108

Figure 1

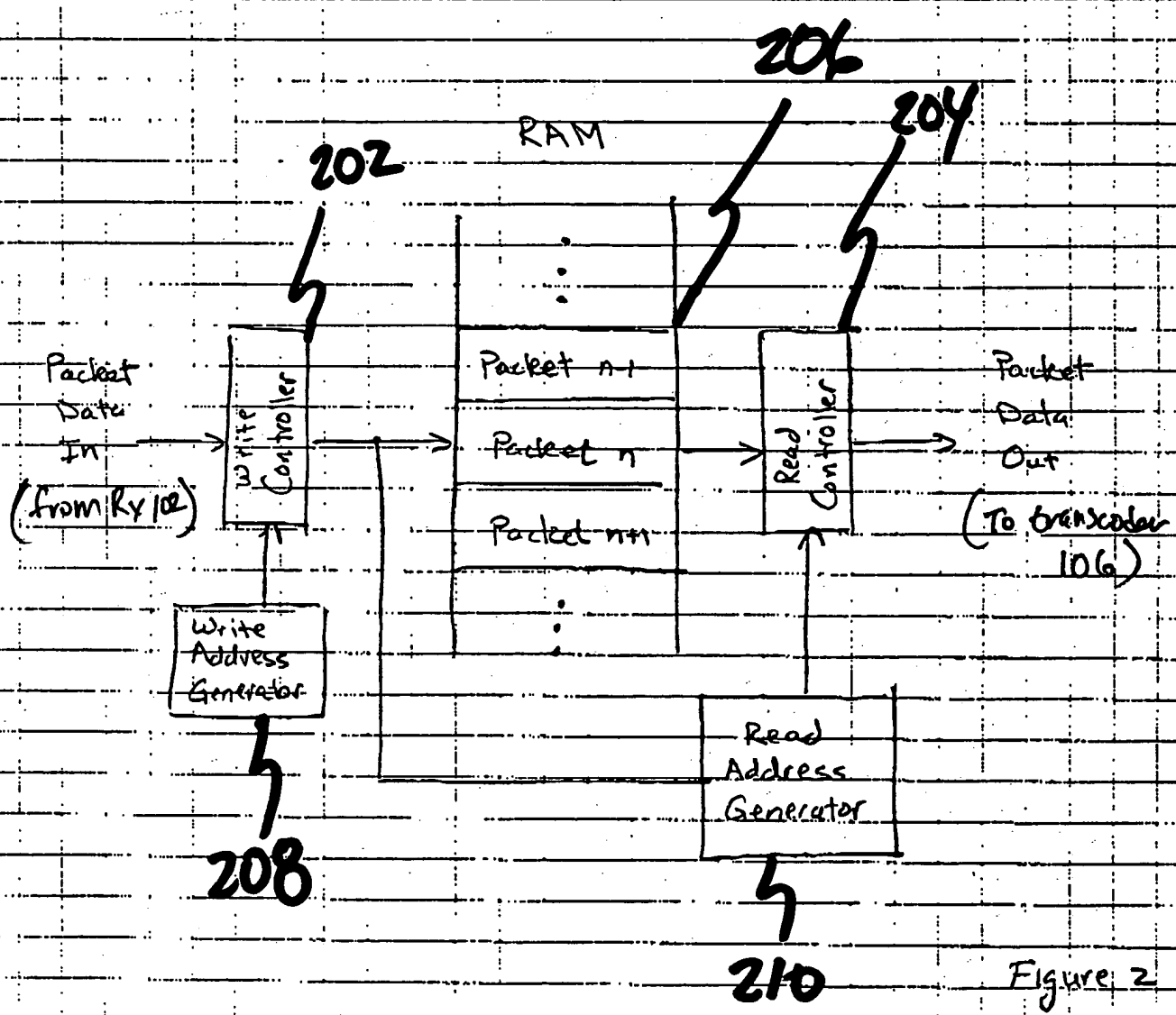

206
RAM
204
202
Packet Data In (from Rx 102)
Write Controller
Packet n-1
Packet n
Packet n+1
Read Controller
Packet Data Out (To transcoder 106)
Write Address Generator
Read Address Generator
208
210

Figure 2

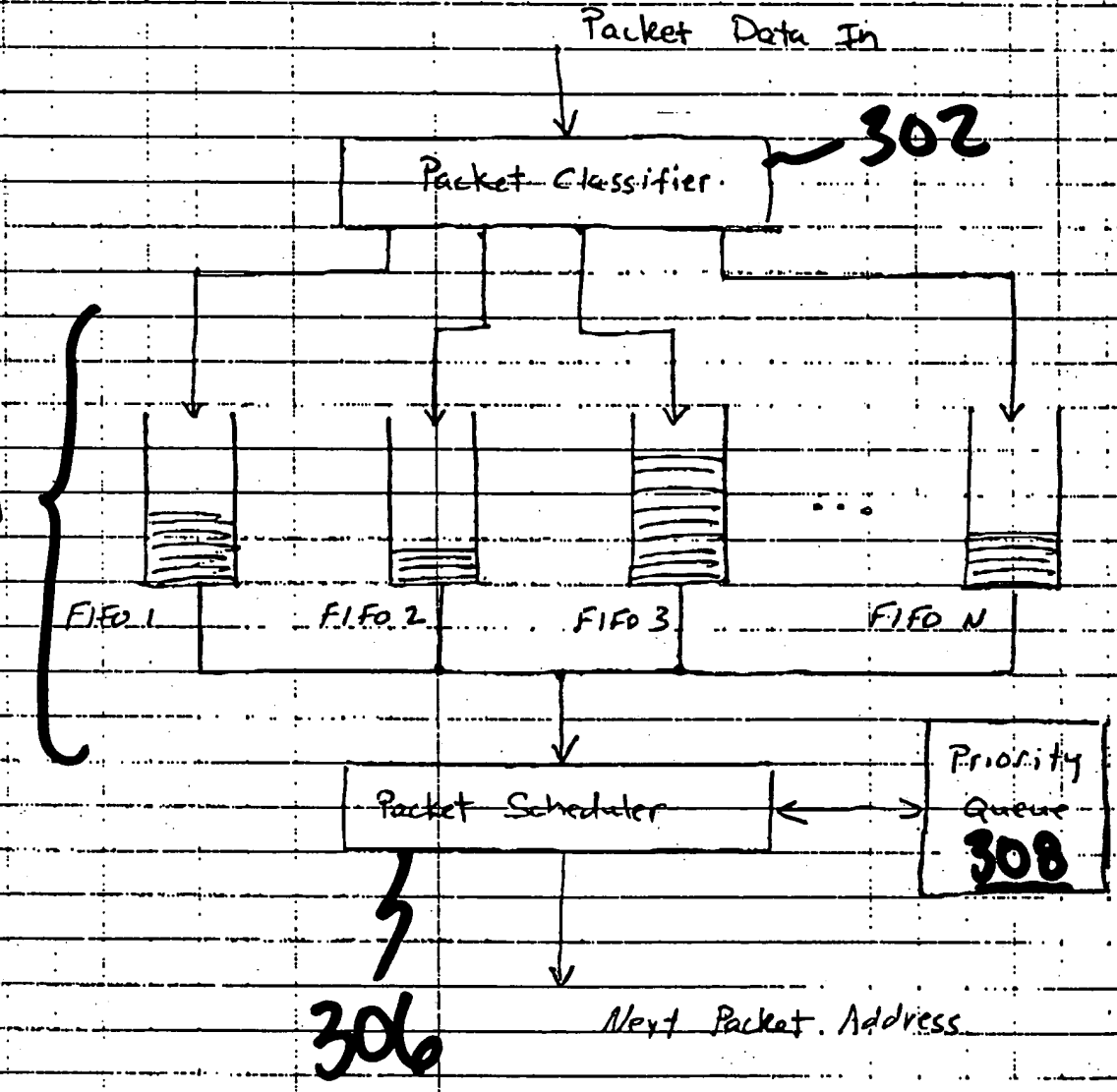
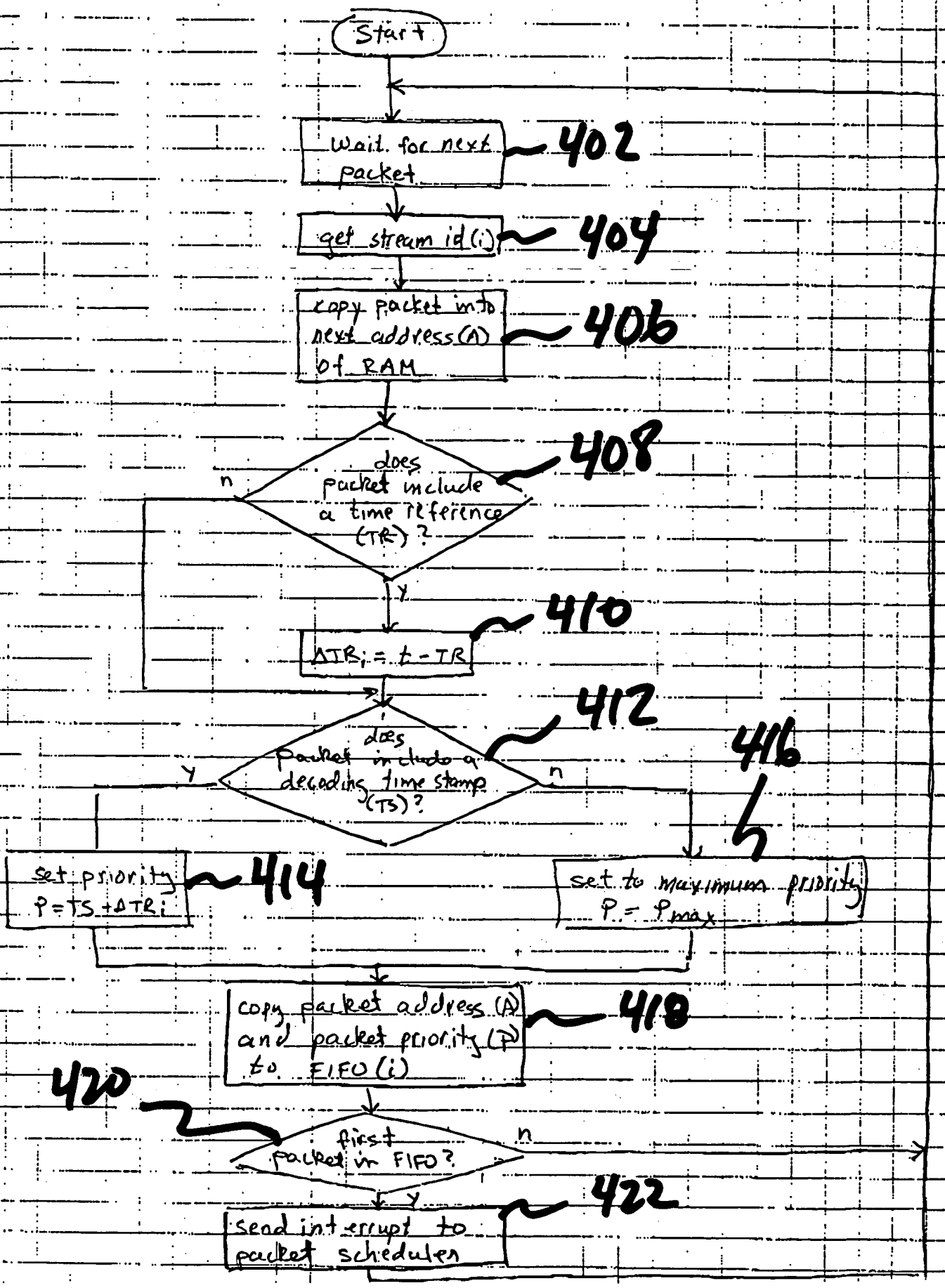

Packet Data In
Packet Classifier
302
304
FIFO 1
FIFO 2
FIFO 3
FIFO N
Packet Scheduler
Priority Queue
308
306
Next Packet Address

Figure 3

Figure 4


Start
Wait for next packet
402
get stream id (i)
404
copy packet into next address (A) of RAM
406
does packet include a time reference (TR)?
408
n
y
ΔTR_i = t - TR
410
does packet include a decoding time stamp (TS)?
412
y
n
set priority P = TS + ΔTR_i
414
set to maximum priority P = P_max
416
copy packet address (A) and packet priority (P) to FIFO (i)
418
420
first packet in FIFO?
n
y
send interrupt to packet scheduler
422

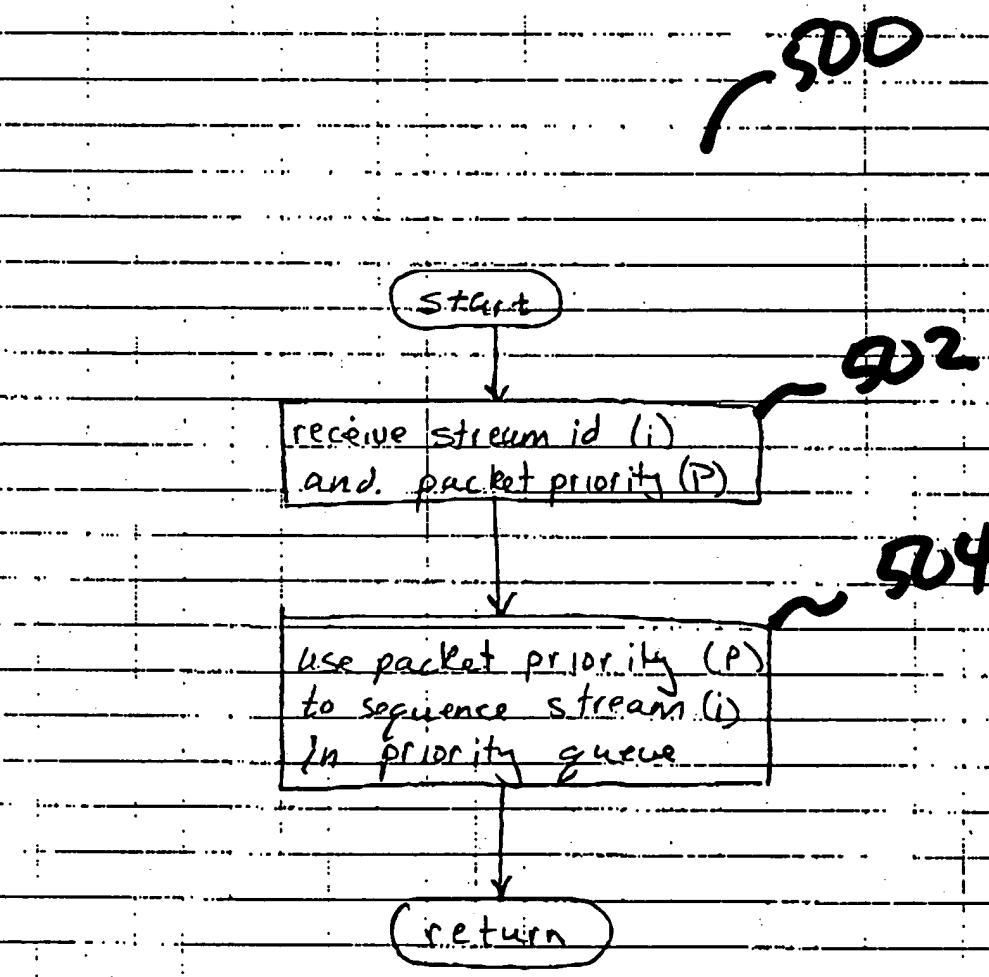

500
start
502
receive stream id (i)
and packet priority (P)
504
use packet priority (P)
to sequence stream (i)
in priority queue
return

Figure 5

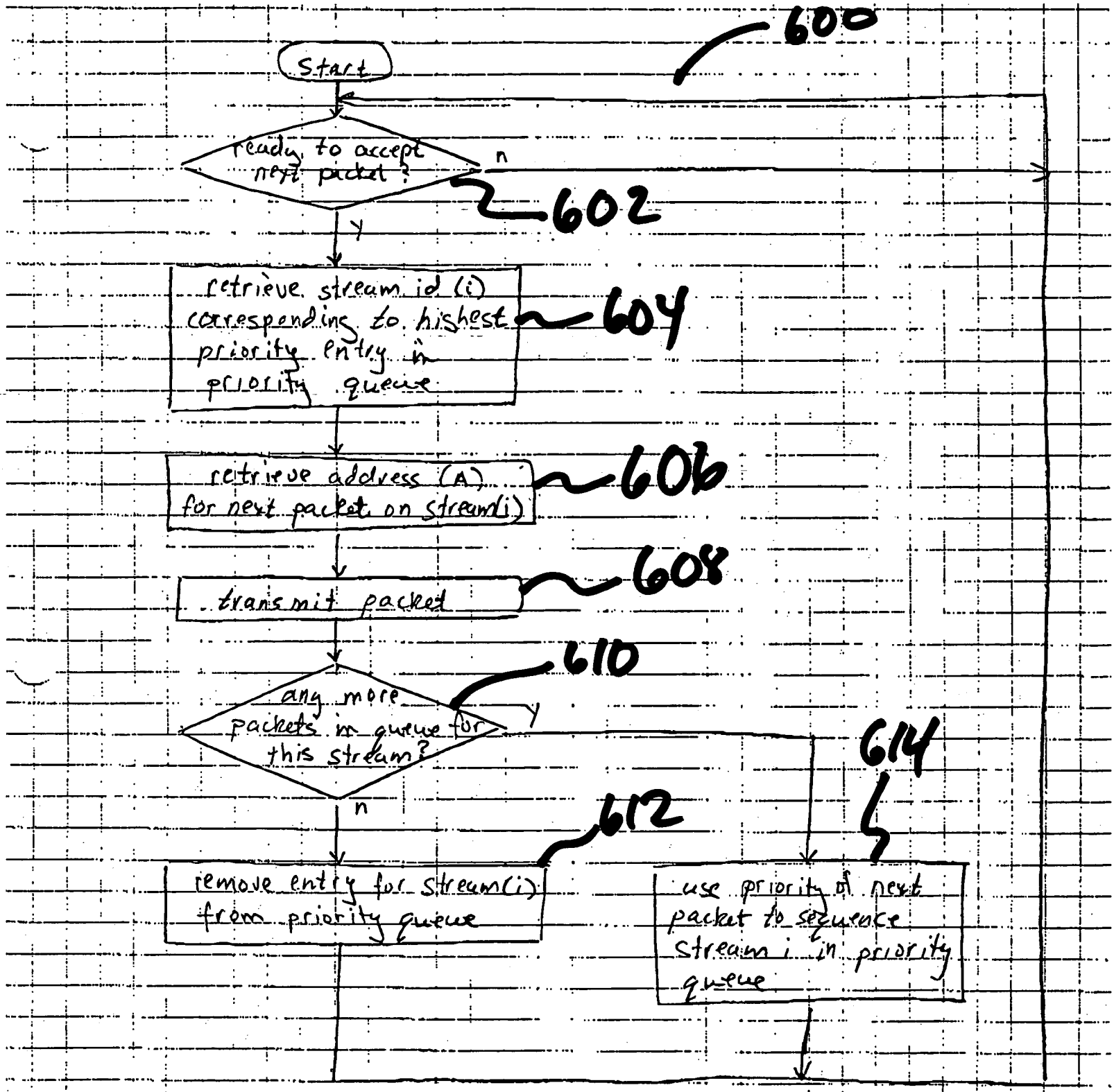
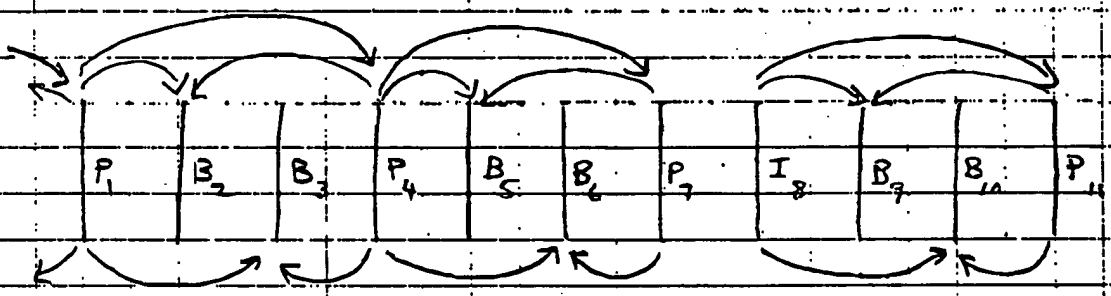

600
Start
ready to accept next packet?
n
602
y
retrieve stream id (i) corresponding to highest priority entry in priority queue
604
retrieve address (A) for next packet on stream(i)
606
transmit packet
608
any more packets in queue for this stream?
610
y
n
614
612
remove entry for stream(i) from priority queue
use priority of next packet to sequence stream i in priority queue

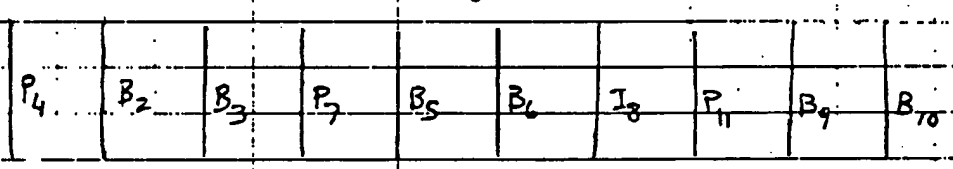
Figure 6

700



Display Order (arrows show direction of prediction)

750



Transmission Order

Figure 7

FROM :

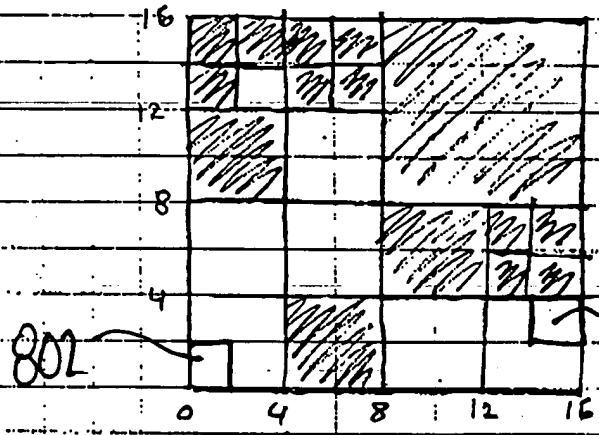


Figure 8A

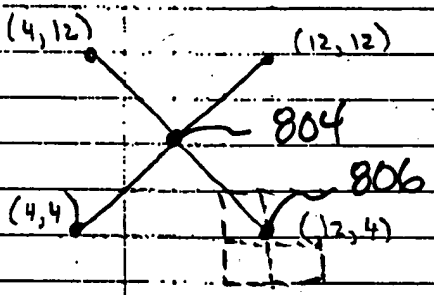


Figure 8B

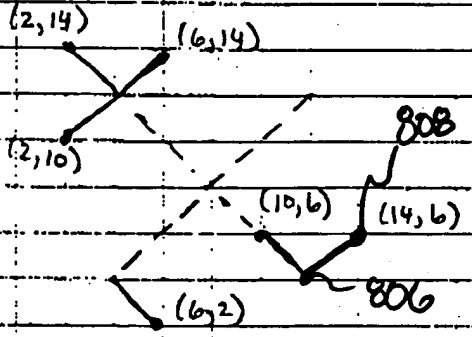


Figure 8C

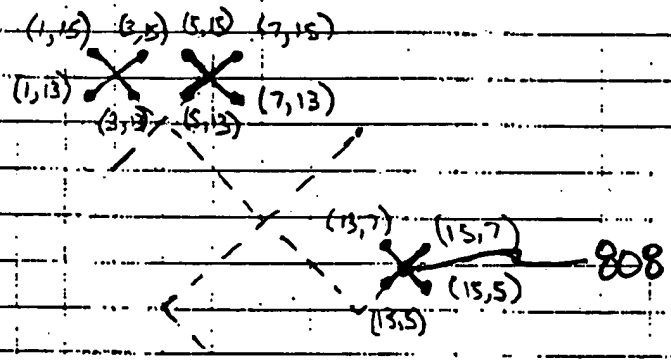


Figure 8D

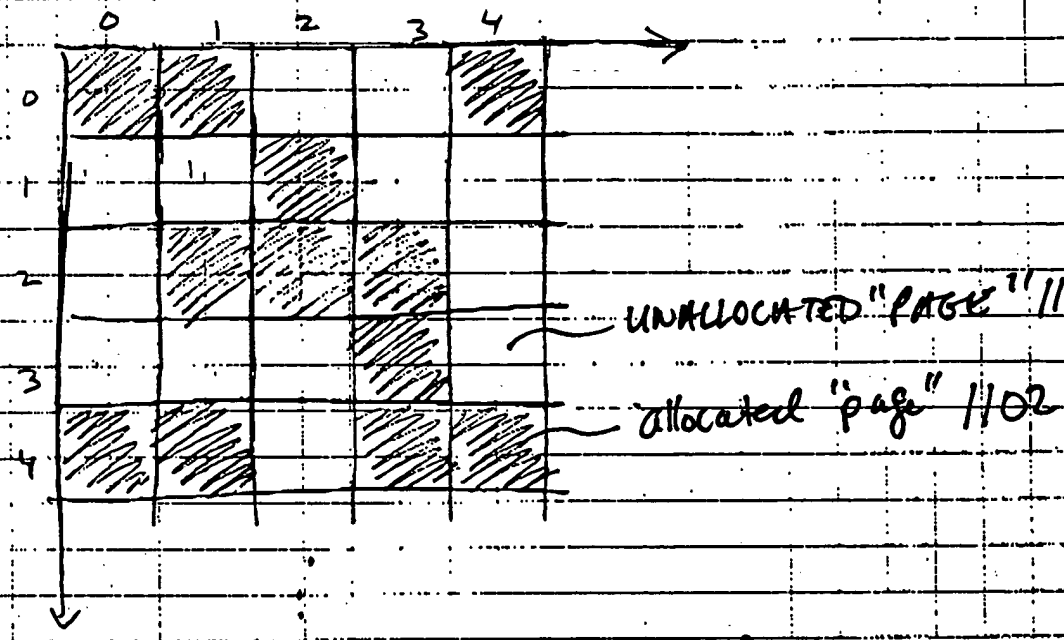
```
mem_allocate ( d, i, j, k ) begin
  if ( d > k) begin
    D(i,j) = 0
    return(addr(i,j))
  end
  k = k / 2
  if ( d <= D(i+k,j+k) and
     ( d  > D(i+k,j-k) or D(i+k,j-k) >= D(i+k,j+k)) and
     ( d  > D(i-k,j+k) or D(i-k,j+k) >= D(i+k,j+k)) and
     ( d  > D(i-k,j-k) or D(i-k,j-k) >= D(i+k,j+k)))
       a = mem_allocate( d, i+k, j+k, k)
  else if ( d <= D(i+k,j-k) and
          ( d  > D(i-k,j+k) or D(i-k,j-k) >= D(i+k,j-k)) and
          ( d  > D(i-k,j-k) or D(i-k,j-k) >= D(i+k,j-k)))
       a = mem_allocate( d, i+k, j-k, k)
  else if ( d <= D(i-k,j+k) and
          ( d  > D(i-k,j-k) or D(i-k,j-k) >= D(i-k,j+k)))
       a = mem_allocate( d, i-k, j+k, k)
  else
       a = mem_allocate( d, i-k, j-k, k)
  D(i,j) = max( D(i+k,j+k), D(i+k,j-k), D(i-k,j+k), D(i-k,j-k))
  r turn( a )
end
```

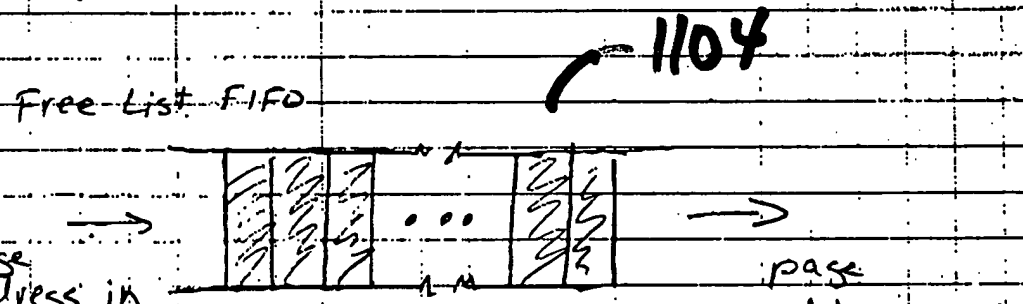
Figure 9

```
mem_free ( i, j, k ) begin
  D(i,j) = 2 * k
  while ( k < MEMSIZE/2 ) begin
    k = k * 2
    D(i,j) = max( D(i+k,j+k), D(i+k,j-k), D(i-k,j+k), D(i-k,j-k))
  end
end
```

Figure 10

Map of Allocated and Unallocated Memory Units





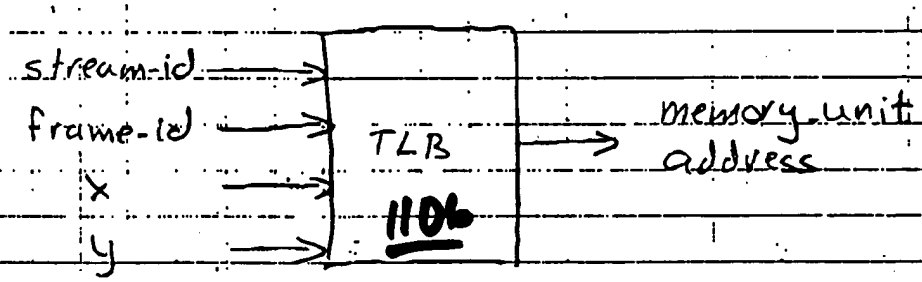


Figure 11

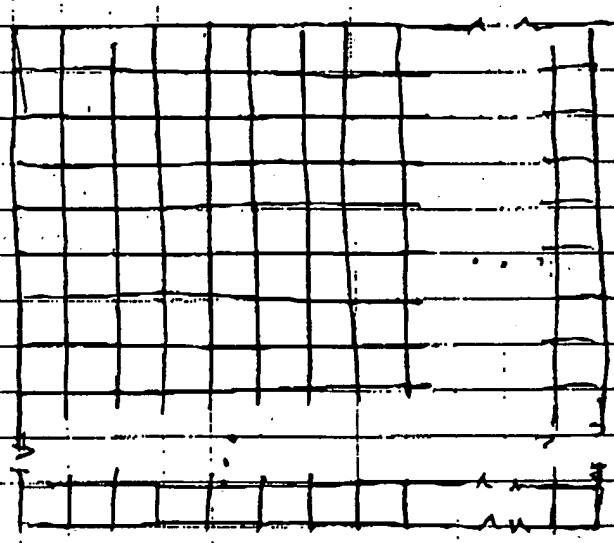


Figure 12

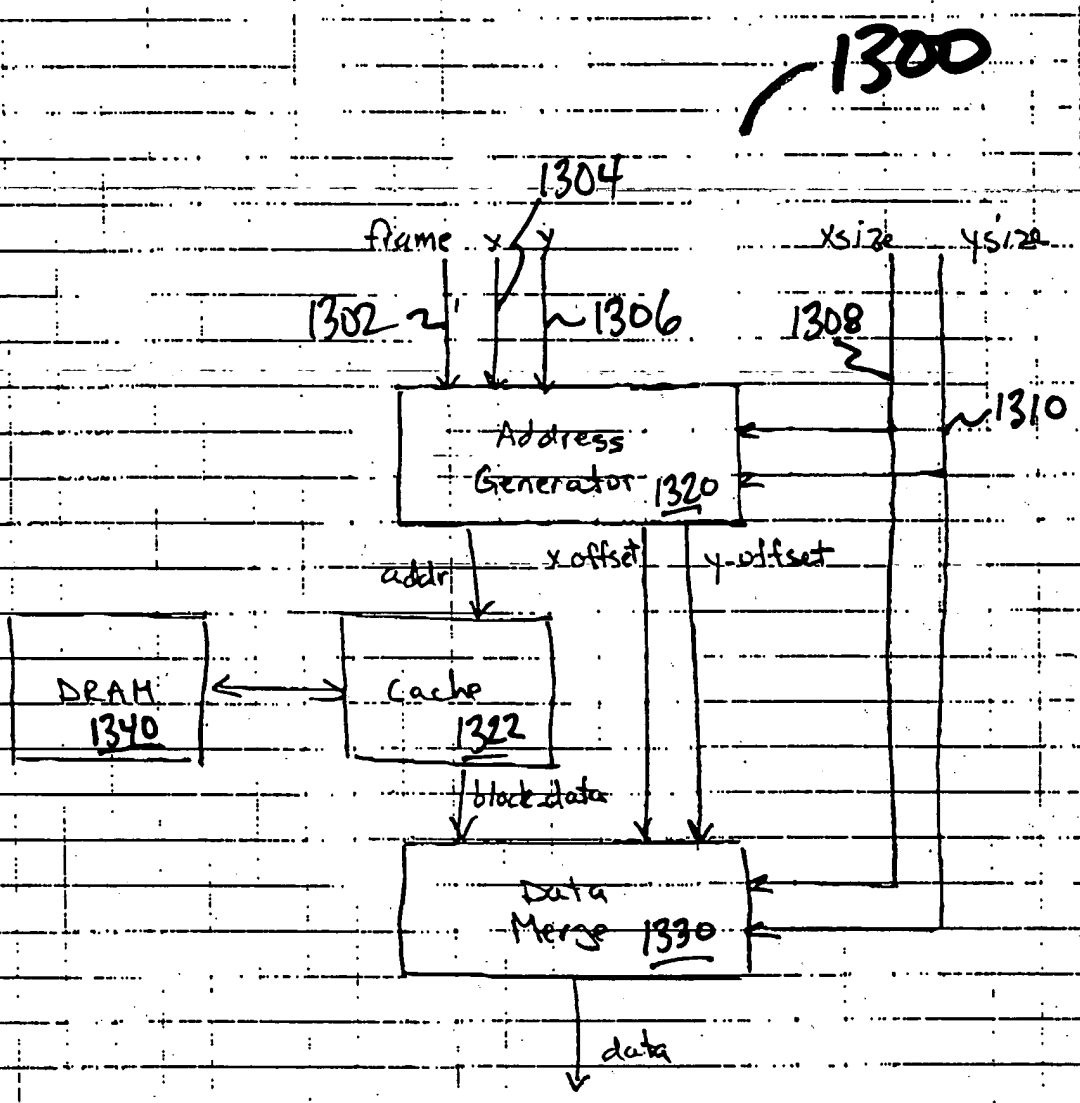

1300
1304
frame
x
y
Xsize
Ysize
1302
1306
1308
1310
Address Generator 1320
addr
x offset
y offset
DRAM 1340
Cache 1322
block data
Data Merge 1330
data

Figure 13

```
addr ss_generator () begin
  m = 0
  n = 0
  input ( frame, x, y, xsize, ysize)
  while (n < ysize) begin
    x = xaddr + m
    y = yaddr + n
    block_addr = LUT { frame, y[:7], x[:7] }
    y_suboffset = y[6:4]
    x_suboffset = x[6:4]
    addr = { block_addr, y_suboffset, x_suboffset }
    y_offset = y[3:0]
    x_offset = x[3:0]
    output ( addr, y_offset, x_offset )
    m = m + 16
    if ( m >= xsize ) begin
      n = n + 16
      m = 0
    end
  end
  return
end

data_merge () begin
  input ( x_size, y_size, x_offset, y_offset)
  n = 0
  while ( n < y_size ) begin
    i = 0
```

Figure 14

```
    while ( i < 16 ) begin
      m = 0
      while ( m < (x_offset + x_size) ) begin
        j = 0
        while (j < 16) begin
          input ( block_data )
          B[i][m] = block_data
          m = m + 1
          j = j + 1
        end
      end
      i = i + 1
    end
    if ( y_offset > 0 ) begin
      i = y_offset
      y_offset = 0
    else
      i = 0
    end

    while ( i < 16 and n < ysize) begin
      while ( j < x_size ) begin
        data = B[i][j + x_offset]
        output ( data )
        j = j + 1
      end
      i = i + 1
      n = n + 1
    end
  end
end
```

Figure 15

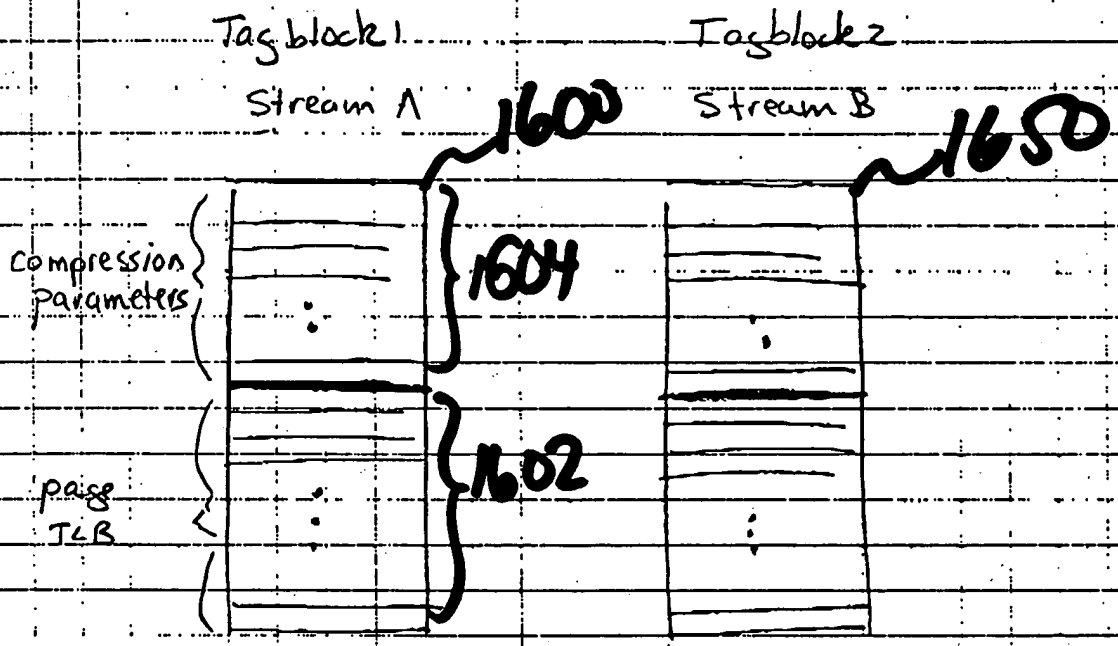

Tagblock1
Stream A
1600
Tagblock2
Stream B
1650
Compression parameters
1604
page TLB
1602

Figure 16

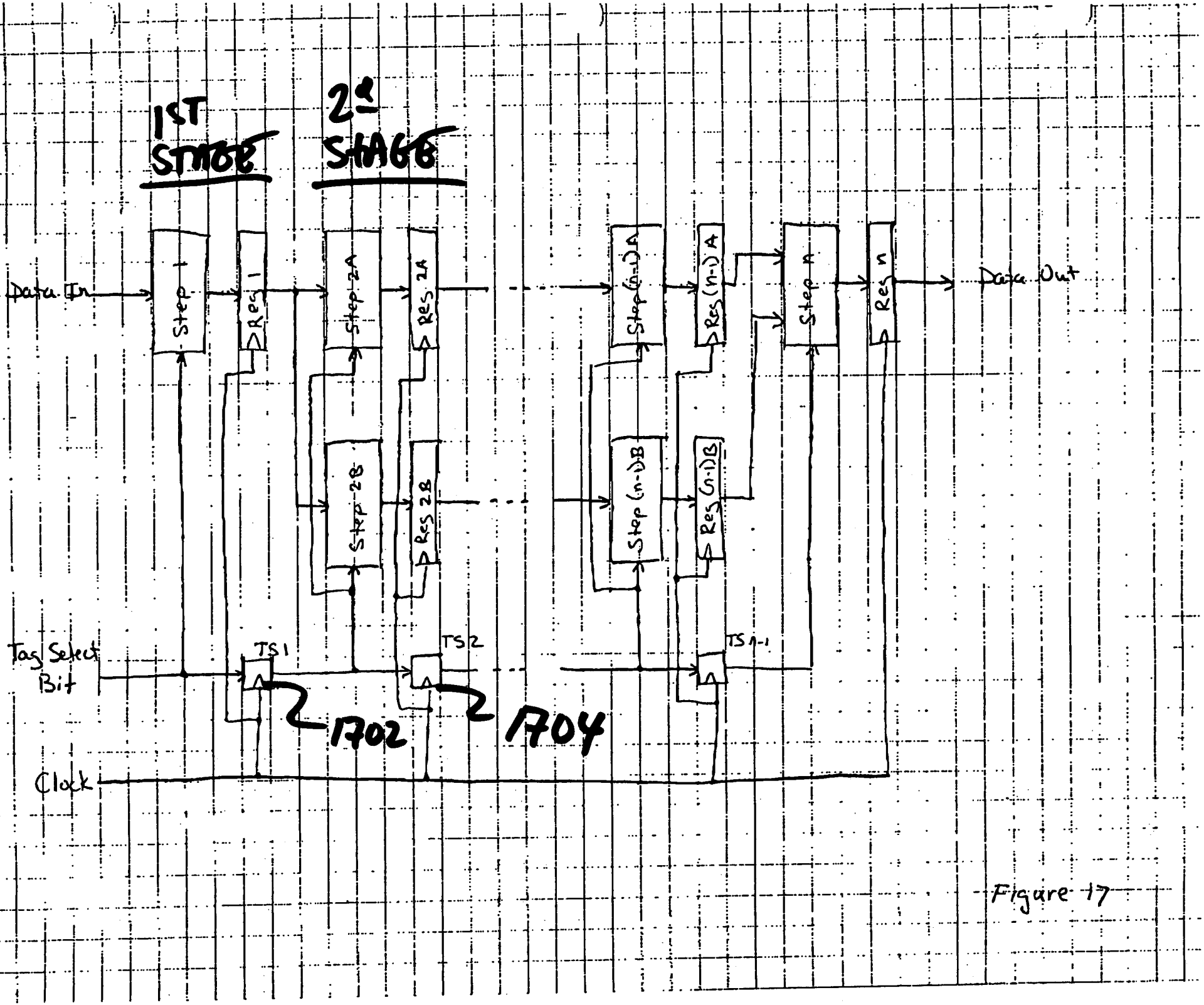

1ST STAGE
2ND STAGE
Data In
Step 1
Res 1
Step 2A
Res 2A
Step 2B
Res 2B
Step (n-1)A
Res (n-1) A
Step (n-1)B
Res (n-1)B
Step n
Res n
Data Out
Tag Select Bit
TS1
TS2
TS n-1
1702
1704
Clock

Figure 17